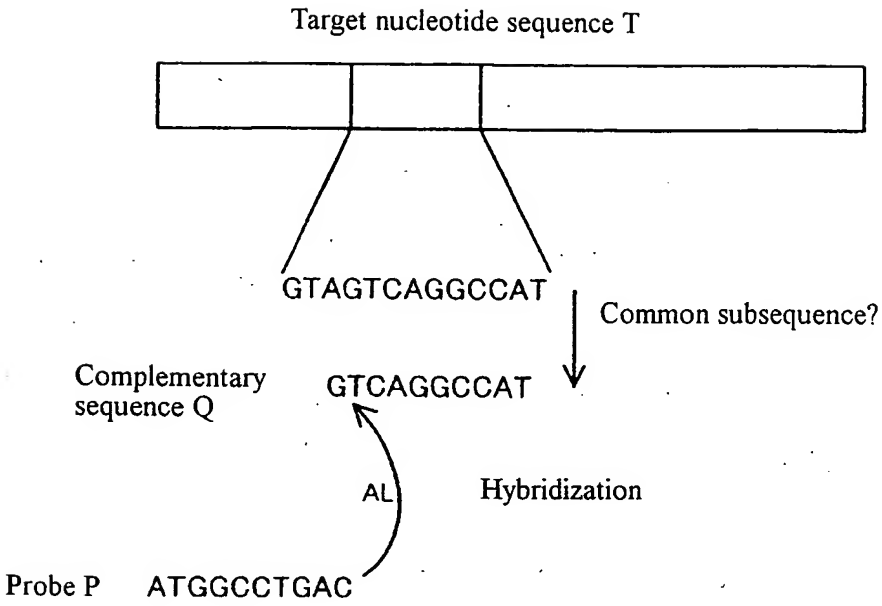
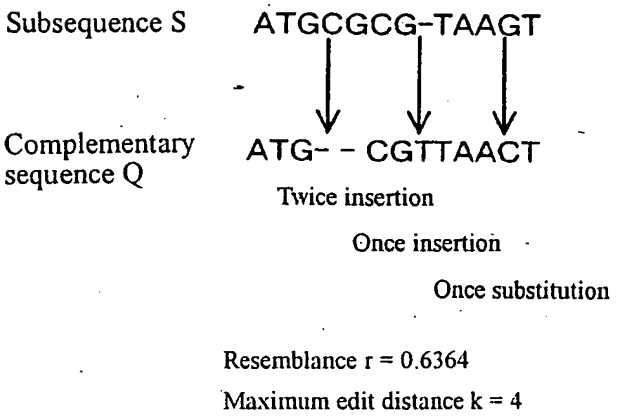

Target nucleotide sequence T
GTAGTCAGGCCAT
Common subsequence?
Complementary sequence Q
GTCAGGCCAT
AL
Hybridization
Probe P
ATGGCCTGAC

(a)


Subsequence S
ATGCGCG-TAAGT
Complementary sequence Q
ATG- - CGTTAACT
Twice insertion
Once insertion
Once substitution
Resemblance r = 0.6364
Maximum edit distance k = 4

(b)

Figure 2

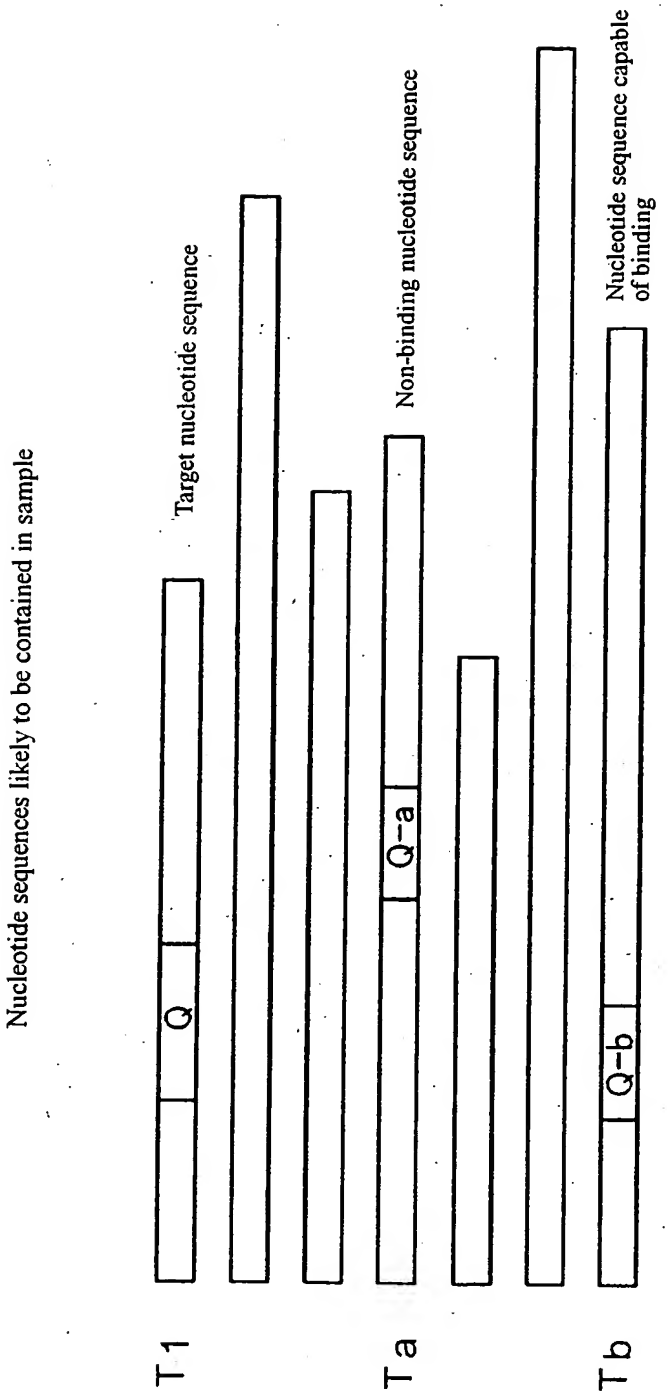
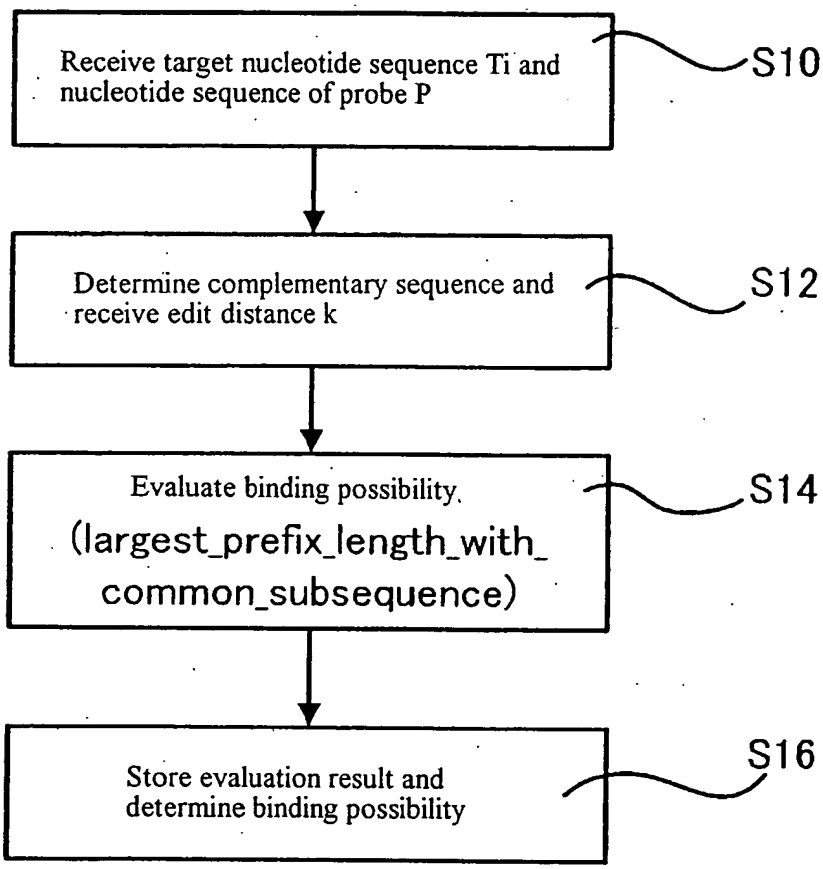

Nucleotide sequences likely to be contained in sample
Target nucleotide sequence
Non-binding nucleotide sequence
Nucleotide sequence capable of binding
Q
Q-a
Q-b
T 1
T a
T b

Figure 3


Receive target nucleotide sequence Ti and nucleotide sequence of probe P
S10
Determine complementary sequence and receive edit distance k
S12
Evaluate binding possibility. (largest_prefix_length_with_common_subsequence)
S14
Store evaluation result and determine binding possibility
S16

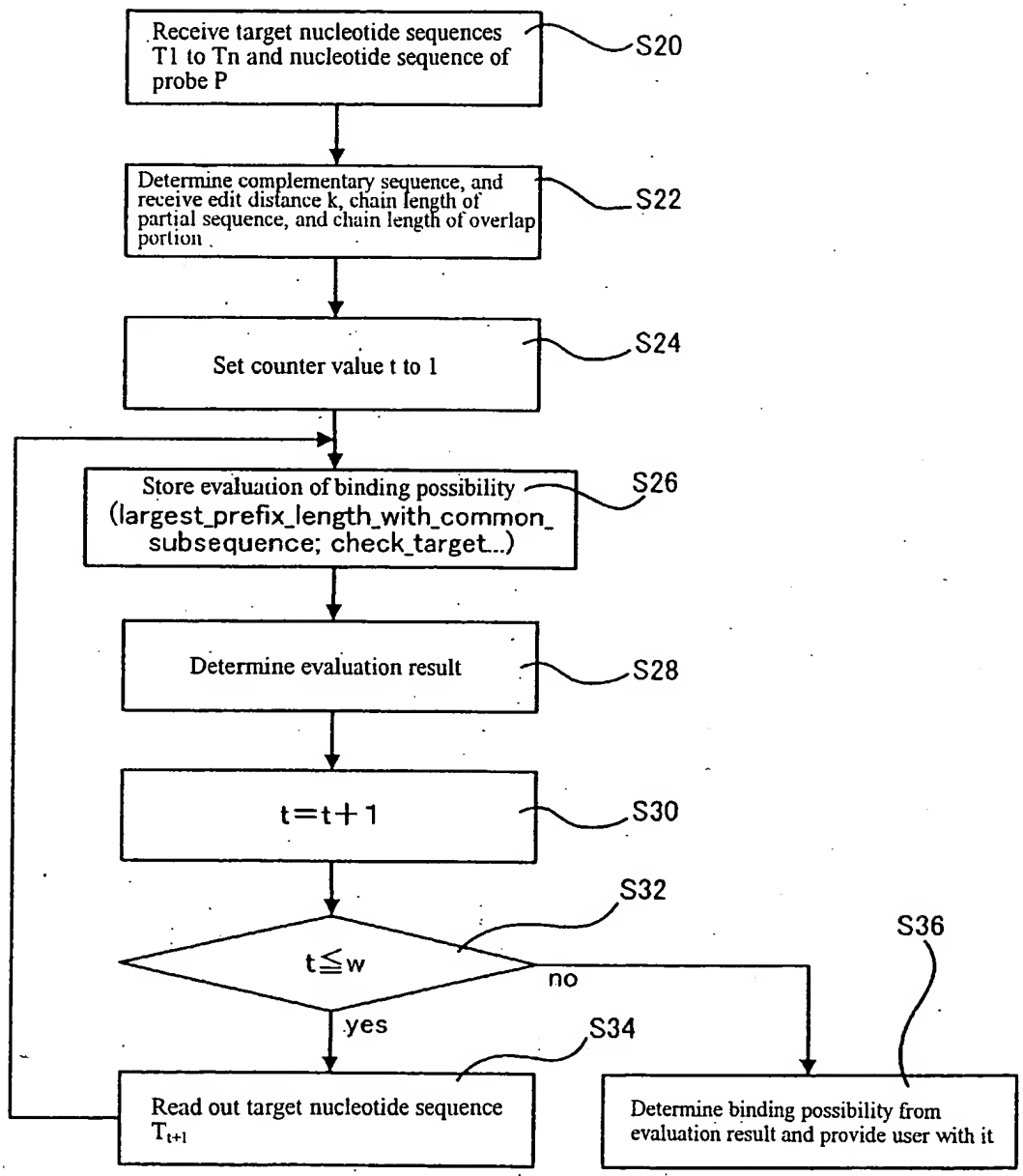

Receive target nucleotide sequences T1 to Tn and nucleotide sequence of probe P
S20
Determine complementary sequence, and receive edit distance k, chain length of partial sequence, and chain length of overlap portion
S22
Set counter value t to 1
S24
Store evaluation of binding possibility (largest_prefix_length_with_common_subsequence; check_target...)
S26
Determine evaluation result
S28
t=t+1
S30
S32
t≦w
no
yes
S34
S36
Read out target nucleotide sequence T_{t+1}
Determine binding possibility from evaluation result and provide user with it

Figure 5

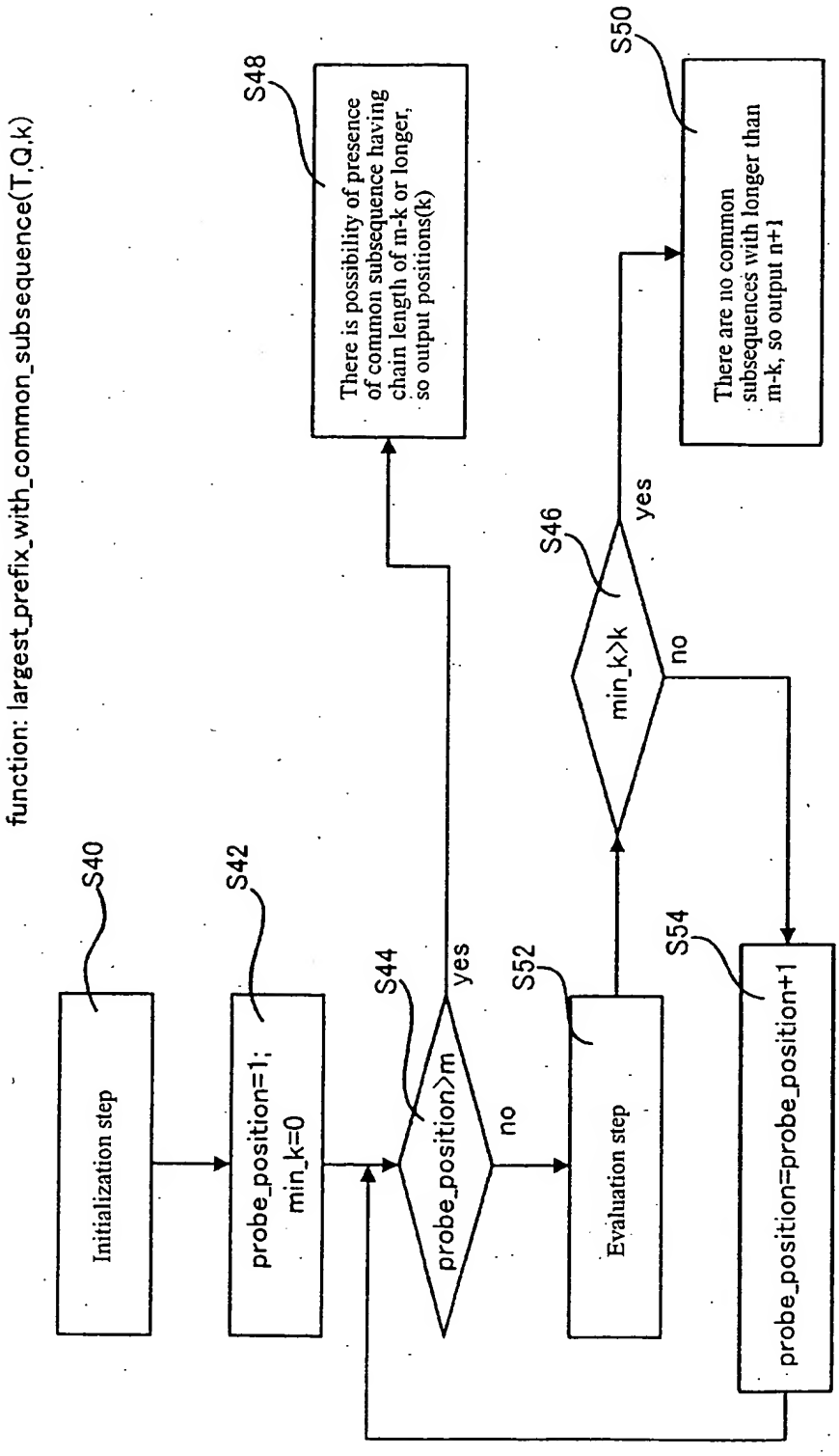

function: largest_prefix_with_common_subsequence(T,Q,k)
S40
Initialization step
S42
probe_position=1;
min_k=0
S44
probe_position>m
yes
no
S48
There is possibility of presence of common subsequence having chain length of m-k or longer, so output positions(k)
S52
Evaluation step
S46
min_k>k
yes
no
S50
There are no common subsequences with longer than m-k, so output n+1
S54
probe_position=probe_position+1

Figure 6

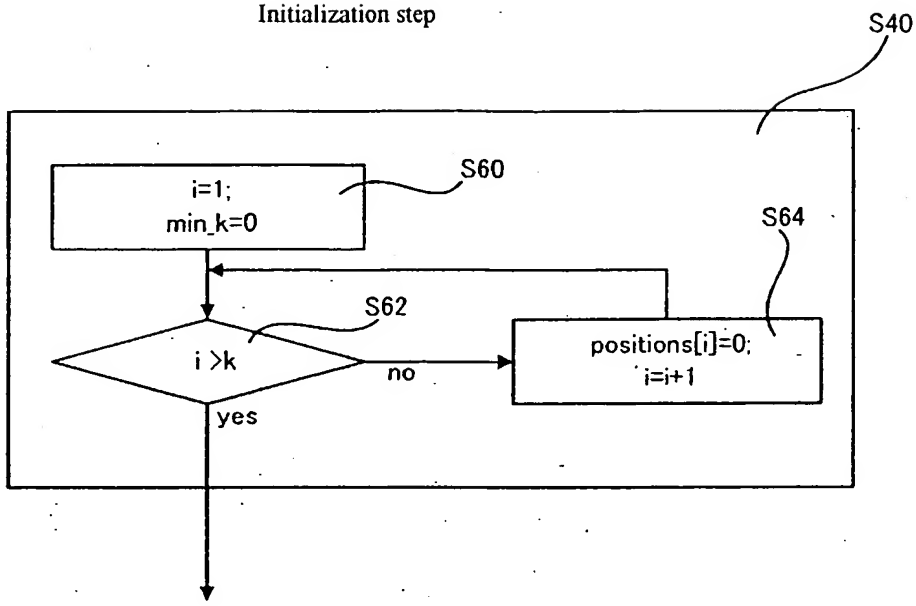
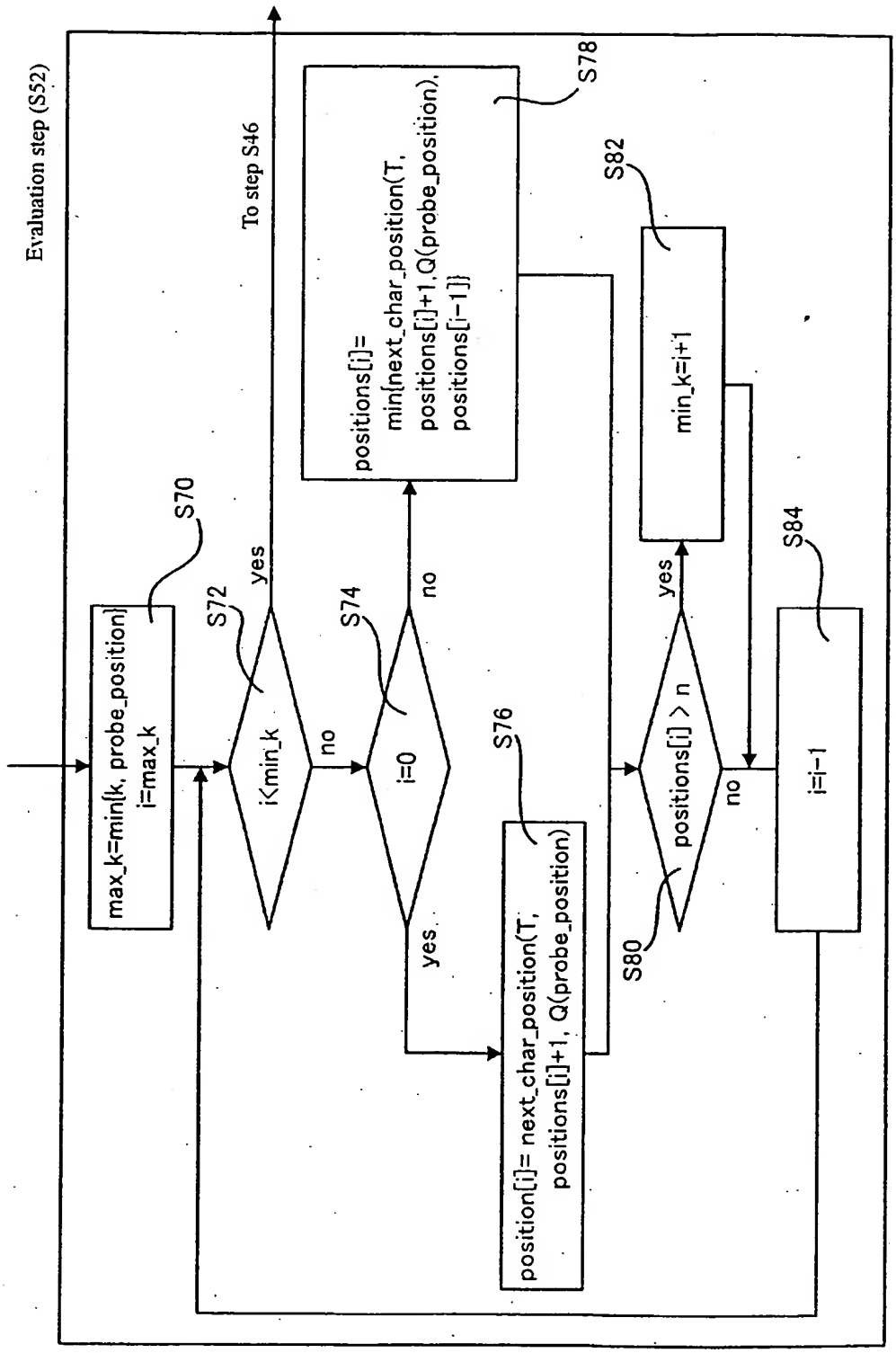
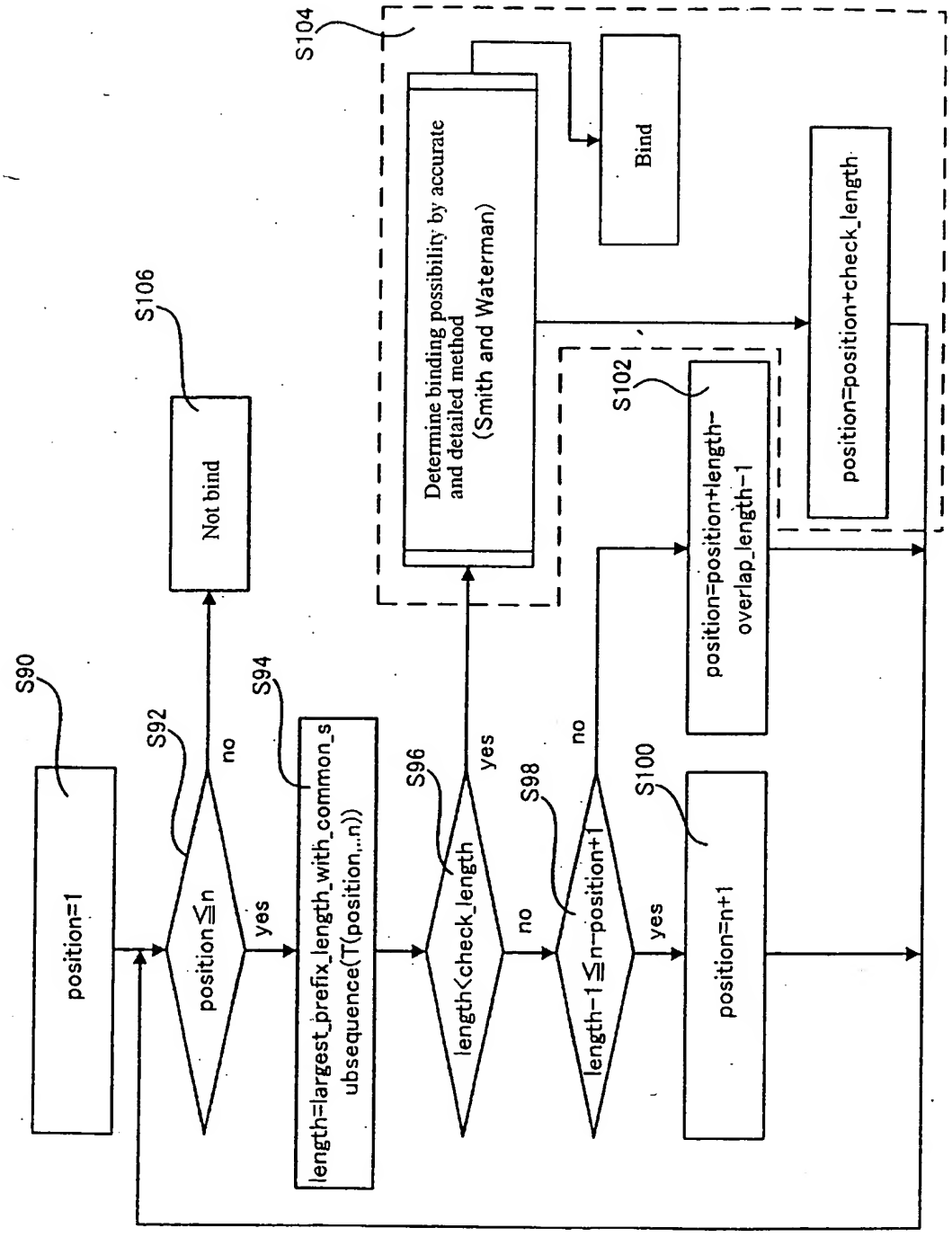

Initialization step
S40
i=1;
min_k=0
S60
S64
S62
i >k
no
positions[i]=0;
i=i+1
yes

Figure 7


Evaluation step (S52)
To step S46
max_k=min(k, probe_position)
i=max_k
S70
i<min_k
S72
yes
no
i=0
S74
yes
no
positions[i]=
min(next_char_position(T,
positions[i]+1,Q(probe_position),
positions[i-1])
S78
position[i]= next_char_position(T;
positions[i]+1, Q(probe_position)
S76
positions[i] > n
S80
yes
no
min_k=i+1
S82
i=i-1
S84

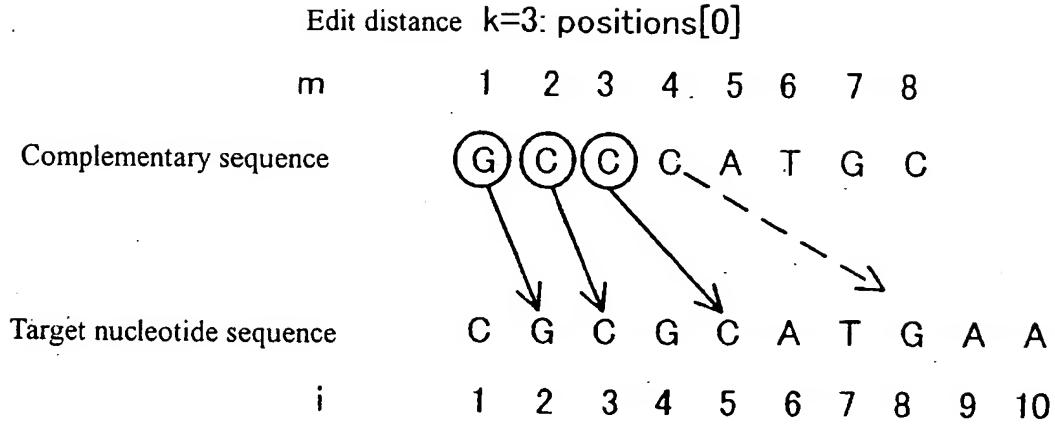

S90
position=1
S92
position≦n
no
yes
S94
length=largest_prefix_length_with_common_subsequence(T(position,,,n))
S96
length<check_length
yes
no
S98
length-1≦n-position+1
no
yes
S100
position=n+1
S102
position=position+length-overlap_length-1
S104
Determine binding possibility by accurate and detailed method (Smith and Waterman)
Bind
position=position+check_length
S106
Not bind

```
//function largest_prefix_with_common_subsequence

largest_prefix_with_common_subsequence(sequence T[1..n], sequence Q[1..m], k) {
    for (i=0;i<=k;i++) positions[i] = 0;
    min_k = 0;
    for (probe_position=1; probe_position<=m; probe_position++) {
        max_k = min {k, probe_position};
        for (i=max_k; i>=min_k ;i--) {
            if (i==0) {
                    positions[i] = next_char_position(T, positions[i]+1,
Q[probe_position]);
            } else {
                    positions[i] = min { next_char_position(T, positions[i]+1,
Q[probe_position]),
                                    positions[i-1] }
            }
            if (positions[i] > n) {
                    min_k = i+1;
            }
        }
        if (min_k > k) {
            ① Absence of common subsequence with length of m-k or longer
            return m+1;
        }
    }
    ② Presence of common subsequence with length of m-min_k or longer
    return positions(k)
```

```
//function check_target

check_target(sequence T[1..n], sequence Q[1..m],k, overlap_length, check_length) {
  position=1;
  while(position <= n) {
    length = largest_prefix_length_with_common_subsequence(
                          T[position..n], Q[1..m], k);
    if (length < check_length) {
        if (check_exactly(
                  T[position..(min(n, position+check_length))], Q, k)) {
            position = position + check_length;
        } else {
          return  ("T binds to P")
        }
    } else if (length -1 <= n - position + 1){
      position = position + length - overlap_length - 1;
    } else {
      position = n+1;
    }
  }
 return ("T does not bind to P");
}
```

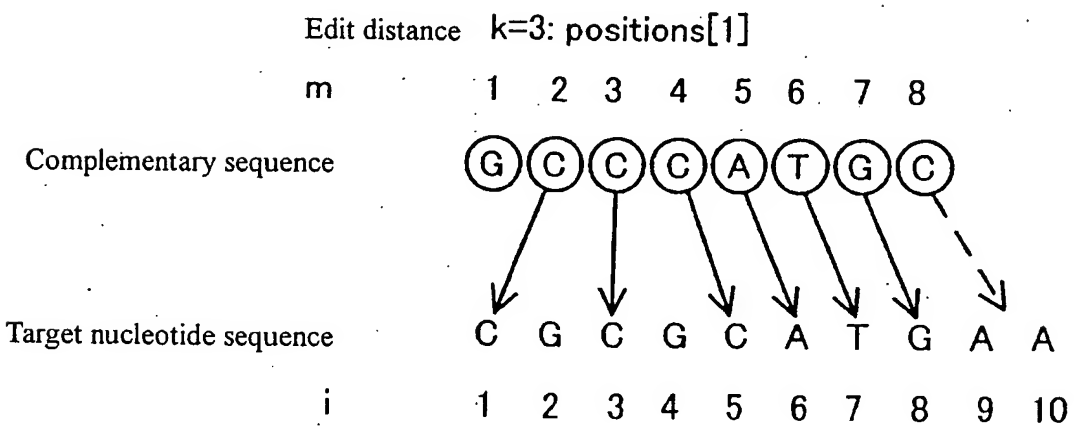
Figure 11

	1	2	3	4	5	6	7	8
positions[0]	2	3	5	-	-	-	-	-
positions[1]	0	1	3	5	6	7	8	-
positions[2]	-	0	1	3	5	6	7	8
positions[3]	-	-	0	1	3	5	6	7



[positions[0],m]= (positions[0], 2, 3, 5, –, –, –, –, –)

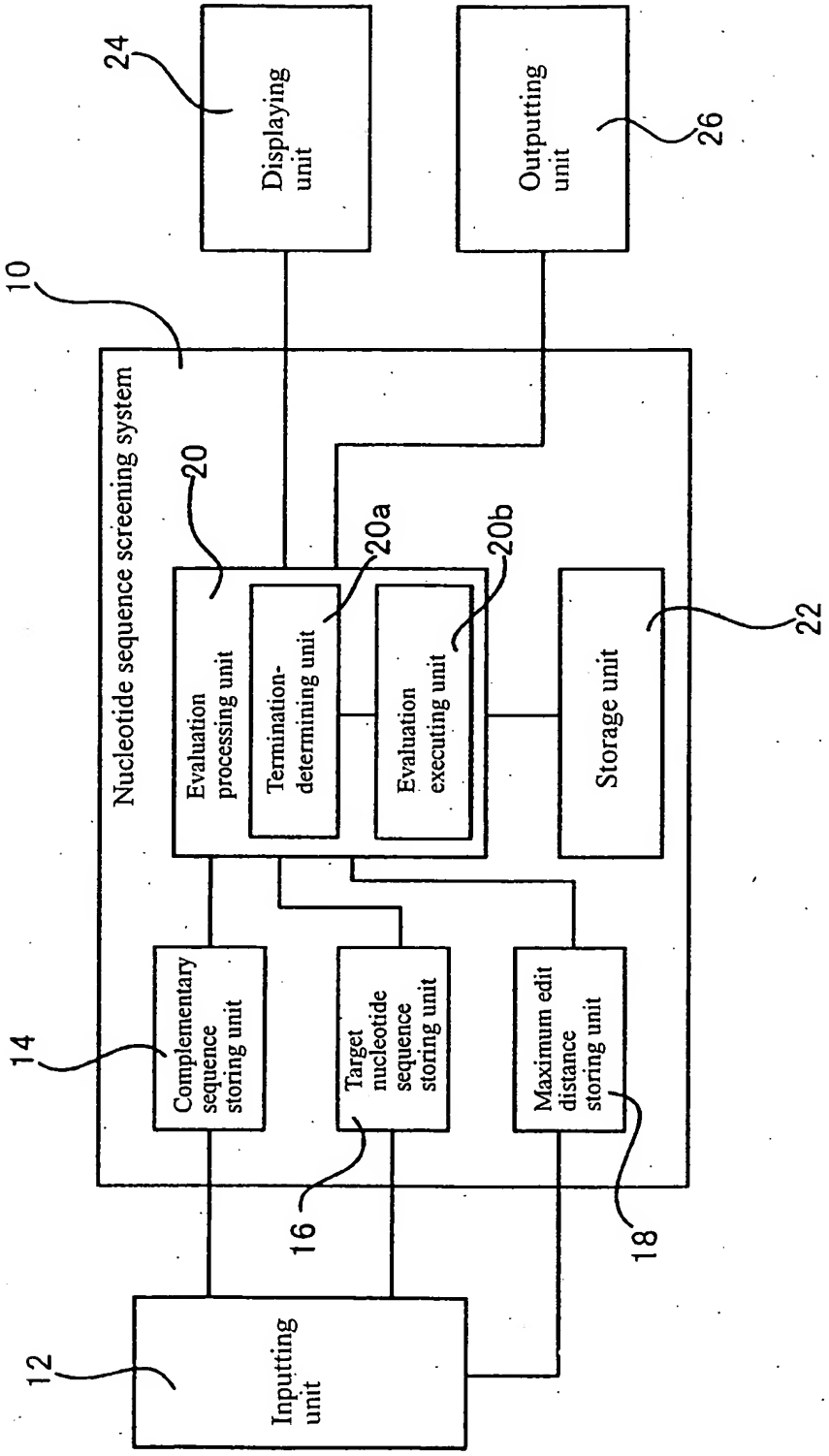
(a)



[positions[1],m]= (positions[1], 1, 3, 5, 6, 7, 8)

(b)

Figure 13


24
Displaying unit
Outputting unit
26
10
Nucleotide sequence screening system
20
20a
20b
Evaluation processing unit
Termination-determining unit
Evaluation executing unit
Storage unit
22
Complementary sequence storing unit
Target nucleotide sequence storing unit
Maximum edit distance storing unit
14
16
18
Inputting unit
12

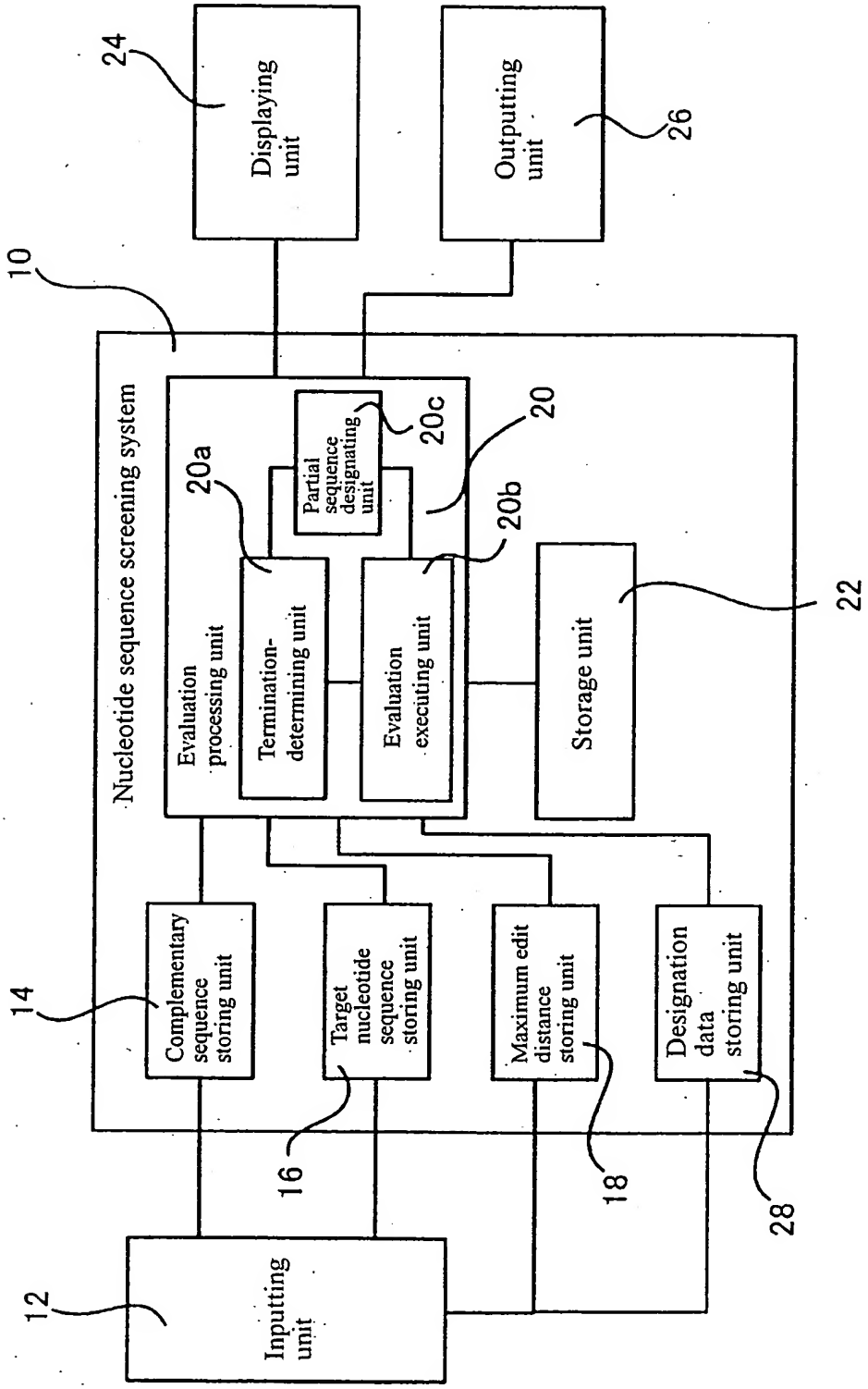

24
Displaying unit
Outputting unit
26
10
Nucleotide sequence screening system
Evaluation processing unit
20a
Termination-determining unit
Partial sequence designating unit
20c
20
20b
Evaluation executing unit
Storage unit
22
14
Complementary sequence storing unit
Target nucleotide sequence storing unit
Maximum edit distance storing unit
Designation data storing unit
16
18
28
12
Inputting unit

Figure 15

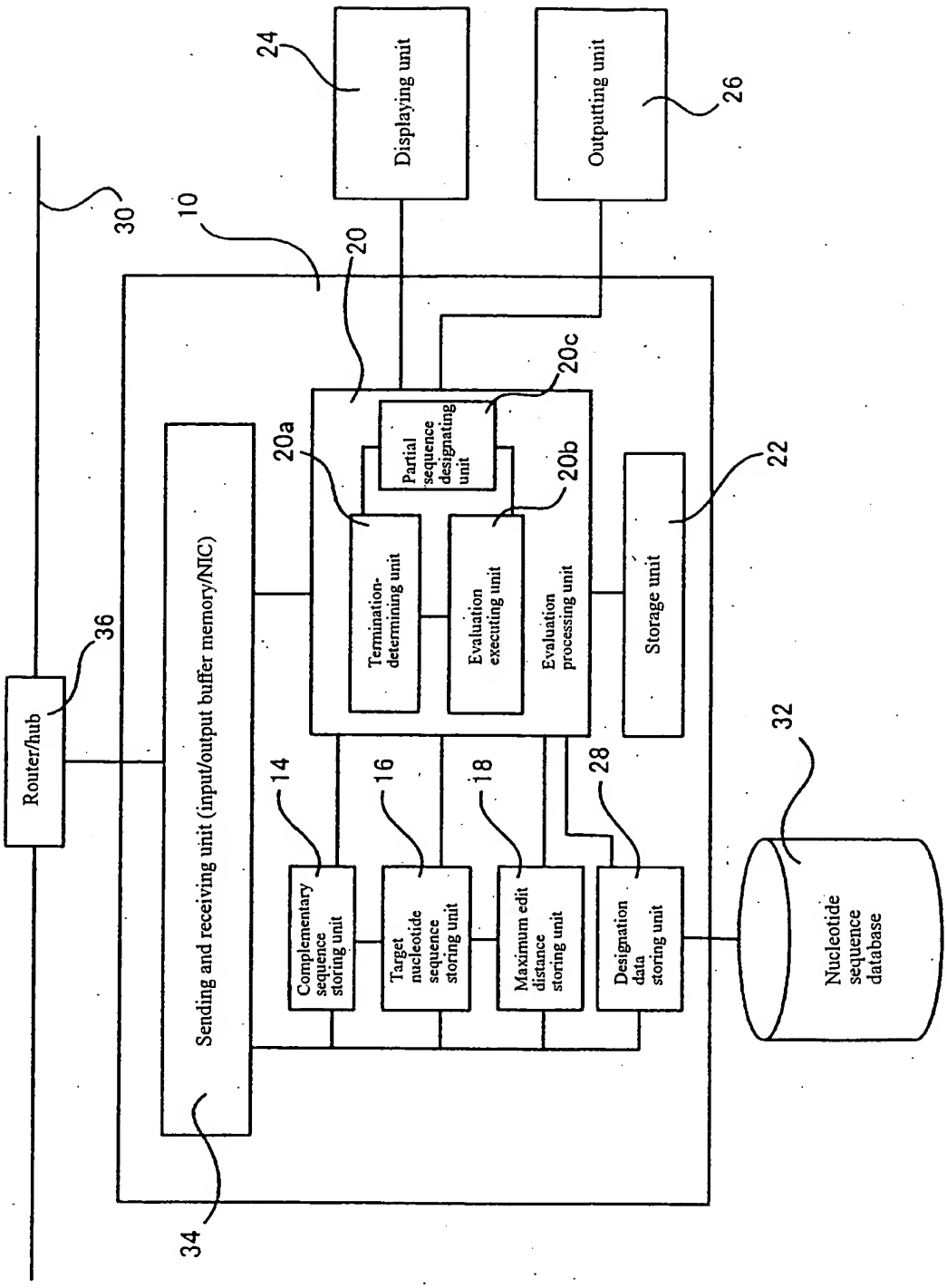

Displaying unit
Outputting unit
24
26
30
10
20
20c
Partial sequence designating unit
20a
20b
22
Termination-determining unit
Evaluation executing unit
Evaluation processing unit
Storage unit
36
Router/hub
Sending and receiving unit (input/output buffer memory/NIC)
14
16
18
28
32
Complementary sequence storing unit
Target nucleotide sequence storing unit
Maximum edit distance storing unit
Designation data storing unit
Nucleotide sequence database
34

Figure 16

Example					Comparative example Smith-Waterman
Resemblance					
0.80	0.85	0.90	0.95	———	
405.47s	284.12s	136.20s	45.05s	1108.33s	

Figure 17

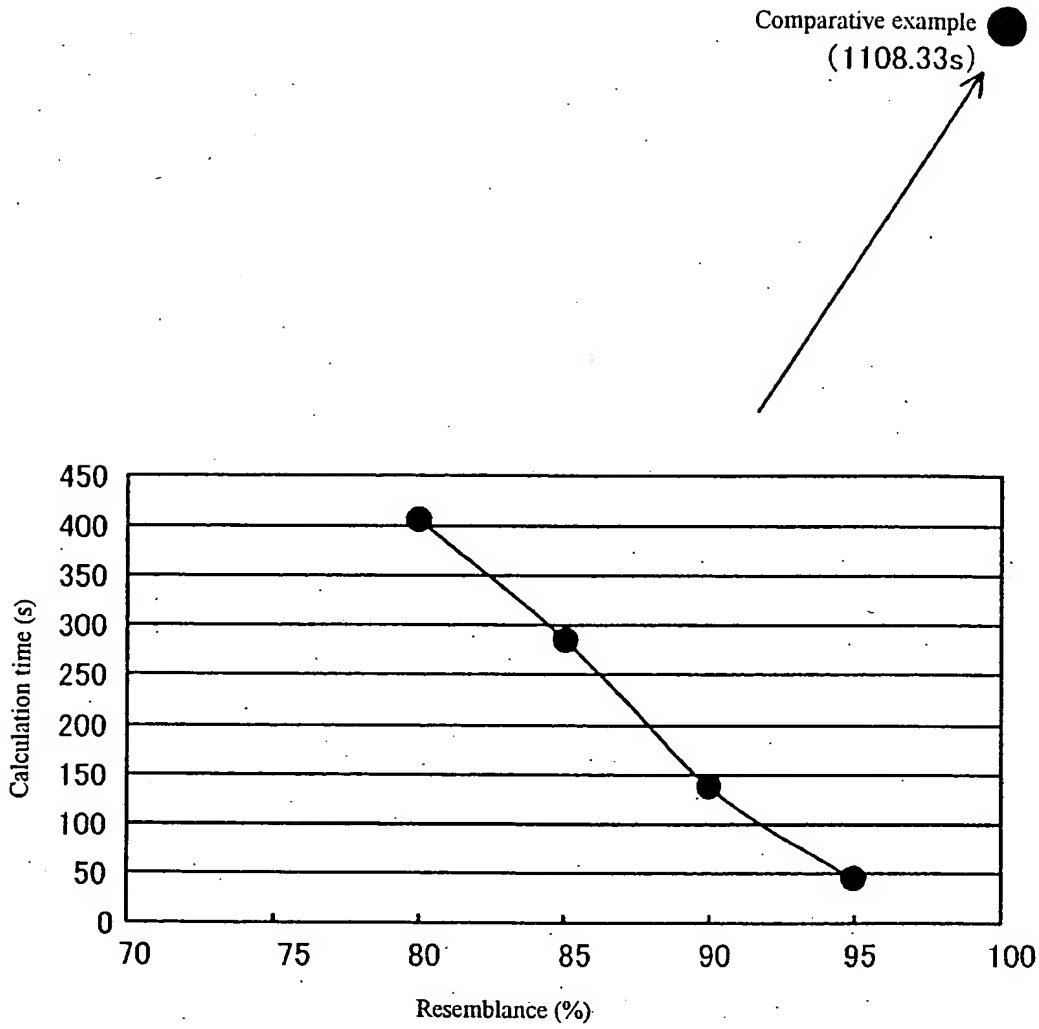
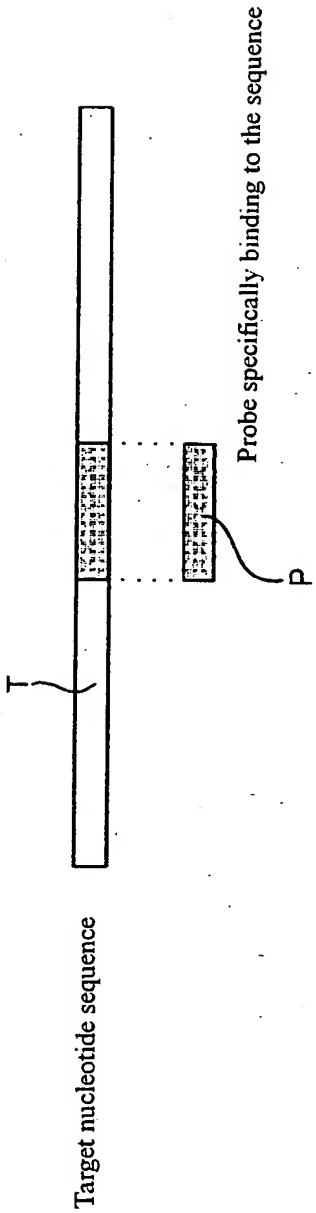

Comparative example
(1108.33s)
450
400
350
300
250
200
150
100
50
0
Calculation time (s)
70
75
80
85
90
95
100
Resemblance (%)

Figure 18


Probe specifically binding to the sequence
P
T
Target nucleotide sequence